**Amendments to the Claims:**

This listing of claims will replace all prior versions, and listings of claims in the application:

**Listing of Claims:**

1. (Currently amended): A computing device that provides hardware conversion of control flow in machine code that is executable by said computing device, said machine code ~~also being executable by a target computing device different from said computing device, said computing device~~ comprising:

predicate assignment means for detecting the beginning and the end of a branch domain of said machine code, operation of said predicate assignment means being invisible to instruction set architecture and thereby invisible to a user; and

predicate use means for realizing the beginning and the end of said branch domain at execution time, and for selectively enabling and disabling machine code within said branch domain during program execution, operation of said predicate use means being invisible to instruction set architecture and thereby invisible to a user,

said machine code being produced by compiling source code which contains at least one conditional branch instruction,

said machine code being executable by a target computing device different from said computing device, ~~wherein~~ said machine code ~~is~~ thereby being executable by said target computing device and by said computing device without recompiling.

2. (Previously presented): The computing device according to claim 1 wherein said predicate assignment means includes a tracking buffer comprising dedicated storage to store branch information in order to make said predicate assignments.

3. (Previously presented): The computing device according to claim 1, wherein said predicate assignment means is operative to assign a canceling predicate to said branch domain in order to delineate said branch domain.

4. (Previously presented): The computing device according to claim 3, wherein said predicate use means further includes dedicated registers for said machine code in order to effect arbitrary control flow, said branch domain including at least a disjoint branch domain, a nested branch domain, overlapped branch domains, or a combination of said branch domains.

5. (Currently amended): A method for providing hardware conversion of control flow to predicates in order to enable a set of machine code comprising a computer program to be executable within a computing device, said set of machine code being executable within a target computing device different from said computing device, said method comprising:

detecting the beginning and the end of a branch domain of selected said machine code in a manner that is invisible to instruction set architecture and thereby is invisible to a user;

generating from each said branch domain a predicate;

associating said predicate with at least one machine code; and thereafter

realizing the beginning and the end of said branch domain at execution time and selectively enabling and disabling execution of machine code within said branch domain

said machine code being produced by compiling source code which contains at least one conditional branch instruction,

said machine code being executable by a target computing device different from said computing device, said machine code thereby being executable by said target computing device and by said computing device without recompiling.

6. (Previously presented): The method according to claim 5 wherein said detecting step includes using a tracking buffer to store branch information to make said predicate assignments.

7. (Previously presented): The method according to claim 5 wherein said predicate generating step assigns a canceling predicate to said branch domain in order to delineate said branch domain.

8. (Previously presented): The method according to claim 7, wherein said predicate generating further includes using dedicated registers for said machine code in order to effect arbitrary control flow, said branch domain including at least a disjoint branch domain, a nested branch domain, overlapped branched domains, or a combination of said branch domains.

9. (Currently amended): A data processor having a first instruction set architecture and configured to execute machine code defined according to a second instruction set architecture different from said first instruction set architecture, the data processor comprising:

first logic to produce domain information indicative of a beginning and an end of a branch domain in said machine code; and

second logic responsive to said domain information to detect a beginning and an end of said branch domain during program execution,

said second logic configured to selectively enable and disable instructions in said branch domain during program execution[[;]],

wherein source code comprising one or more conditional branch instructions is compiled to produce a machine code representation of the source code, said machine code representation being executable by both said data processor and by a target data processor having said second instruction set architecture.

~~wherein said data processor can execute said machine code written for said second instruction set architecture without recompiling said machine code for said first instruction set architecture.~~

10. (Previously presented): The data processor of claim 9 wherein said first logic includes a tracking buffer to store said domain information.

11. (Previously presented): The data processor of claim 9 wherein said domain information comprises an address of a predicate that corresponds to a branch, an address of a canceling predicate that corresponds to said branch, and a target address of said branch.

12. (Previously presented): The data processor of claim 9 wherein said branch domain including at least a disjoint branch domain, a nested branch domain, overlapped branch domains, or a combination of said branch domains.